A Multicollaborative Push-Caching HTTP Protocol for the WWW

Alejandro López-Ortiz - Daniel M. Germán

Abstract:

We propose a caching protocol designed to automatically mirror heavily accessed WWW pages in a distributed and temporal fashion. The proposed caching mechanism differs from proxy type mechanisms in that it caches according to load pattern at the server side, instead of access patterns at the client-side LAN, in a Demand-based Document Dissemination (DDD) system fashion. This type of server initiated caching scheme has been termed push-caching. As well, the proposed caching scheme incorporates topological caching functions. The proposed protocol is orthogonal to other extensions to the HTTP protocol and other caching schemes already in use.

Table of Contents

Introduction and Motivation

The World Wide Web has seen a dramatic increase in popularity during the past sixteen months. As a result of this success, there have been interruptions of service from the part of heavily accessed Web sites, such as Cool Site of the Day, WWW indexing services and other popular servers. It is clear that most organizations do not have the bandwidth or computational resources required to support the heavy loads associated with a popular site.

The Net has been identified as a great resource for the dissemination of information: a printing press on everybody's hands. Ironically, the more popular a personal site is, the likelier it is to be discontinued, due to server load. Only large commercial operators can meet the computing requirements of a frequently accessed site. Distributed methods of information broadcast, such as a posting to an Usenet newsgroup or a radio broadcast have a fix set of demands on the broadcaster which is independent of the number of

people who actually read or listen to the information distributed. On the other hand, other protocols such as WWW and cable TV require additional investment for every new user. In the latter case, costs are simply passed on to the consumer. But as the WWW is based on the free distribution of information, similar cost recovery schemes are not equally feasible.

As well, transmition of information is highly redundant say, as opposed to Usenet. While a posting to a newsgroup essentially travels any given part of the network but once, a WWW page accessed by two users from the same organization at the same time generates two independent transmissions Gwe94, Gwe95]. This duplication of broadcast is not unlike that generated by FTP. Indeed, the Alex caching scheme [Cat92] was designed to alleviate these problems while at the same time providing a more familiar user interface to FTP transmissions. As FTP traffic has not increased rapidly enough to be a threat to network bandwidth, Alex has remained subutilized in spite of its clear benefits. On the other hand, traffic on the WWW is fast reaching the critical point of saturation.

To this regard, the National Science Foundation deemed as a critical research topic for the National Information Infrastructure to ‘‘develop new technologies for organizing cache memories and other buffering schemes to alleviate memory and network latency and increase bandwidth’’ (iNSF94] as quoted in [Bes95]).

Because of these considerations several research groups are studying the impact on traffic by proxy/cache additions to the HTTP protocol. It has been shown that the addition of *organization edge* proxies for incoming traffic, as well as remote cache servers, would result in a noticeable reduction in expected traffic [Bes95].

Currently, some popular browsers, such as Netscape, and some large organizations, such as DEC [Jon94], provide some degree of local caching for their users. As traditionally configured, Netscape retains the last few images and text files accessed in a cache directory which is accessible only to the requestor. This results in a reduction of network traffic and latency. Load in the server, however, is only marginally reduced as the cache is accessible to one user alone. Similarly, the internal DEC network is equipped with a caching relay for the organization (not unlike that of Netscape) to which all internal user requests are first directed. If a hit occurs, the information is served to the user, else the relay host forwards the request to the actual server indicated by the URL. Again, the impact on server load is minor.

At this time, there are proposed modifications to the HTTP protocol, currently implemented in several HTTP servers, thaty make caching possible for pages which are frequently modified. This new command in the protocol, called ‘‘if-modified-since’’ allows a caching client, the so-called proxies, to verify if a cached document has been recently modified. If so, it requests a fresh version of it, otherwise it serves it locally to the user, without generating additional external traffic.

In early August 1994, we started tracing access patterns at a local Web server, and by late October 1994, our measurements confirmed the observations stated in [CBP94]. Thus it became clear that a form of server initiated caching would eventually be necessary. The last fourteen months have, if anything, exceeded our expectations, and more than justify the requirement of server-side caching. However, at this time, there seems to be little work on this area, and none whatsoever at the level of HTTP specifications.

In this work we present a push-caching scheme which, as opposed to [Gwe95], is additional to current client-side proxying schemes.

We propose a collection of collaborative proxies that cooperate in caching of WWW documents. The protocol is designed to implement a Demand-based Document Dissemination system or DDD (as proposed by [Bes95]), meaning that the request for caching is issued by the server depending on local load and size statistics. Among the advantages of a DDD system are that automatic mirroring is provided as well as the efficient use of resources, since documents are cached according to demand and geographic proximity.

A significant difference with other caching schemes is that in this multicollaborative system, caching is initiated by the server. As the server is aware of the modification frequency of a document (say by verifying the last modified date), it avoids many of the problems posed by ‘‘mutable’’ documents in [Bes95] by means of not caching highly mutable documents. (Mutable documents are those which are frequently ‘‘updated’’.) As well, this scheme partially implements the ideas of geographic and topological caching.

In section 2, we define the different types of traffic generated by Web accesses. Section 3 describes the caching scheme and discusses specific issues of its implementation. In section 4 we describe the results of computer modelling of the caching scheme.

Traffic Overload

Traffic generated by WWW transactions can be classified in four different classes.

1. LAN External Traffic
2. LAN Internal Traffic
3. WAN Single Source Traffic
4. WAN Multi Source Traffic

This distinction is very relevant, as different types of solutions are required to reduce different types of traffic. First, let us specify what we mean by each category.

LAN External Traffic is generated by external users accessing locally maintained documents in different internal hosts. Thus, if in the local network configuration the server host A is connected to a host B which in turn is connected to the external Internet gateway server C, each access to a document in A causes network traffic to be increased locally between A, B and C; which in most cases entail a degradation in transmission speed for local users connected to the subnetworks A-B and B-C.

LAN Internal Traffic are local users repeatedly accessing, across an organization’s internal network, an internal or external WWW document. Once again, traffic will be increased on the local subnetworks that contain gateways connecting the client host to the server host.

WAN Single Source Traffic is generated by users from the same first or second level domain, such as a country or organization, accessing the same WWW page. In this case we have an international or backbone wire carrying several copies of the same page within relatively short spans of time (see figure 2).

WAN Multi Source Traffic occurs when a popular WWW page is widely accessed across the globe in such a way that no individual organization generates a significant percentage of the traffic, while at the

same time, traffic is high enough to bring the server to a halt. In this case, the main objective is to reduce server load rather than network load.

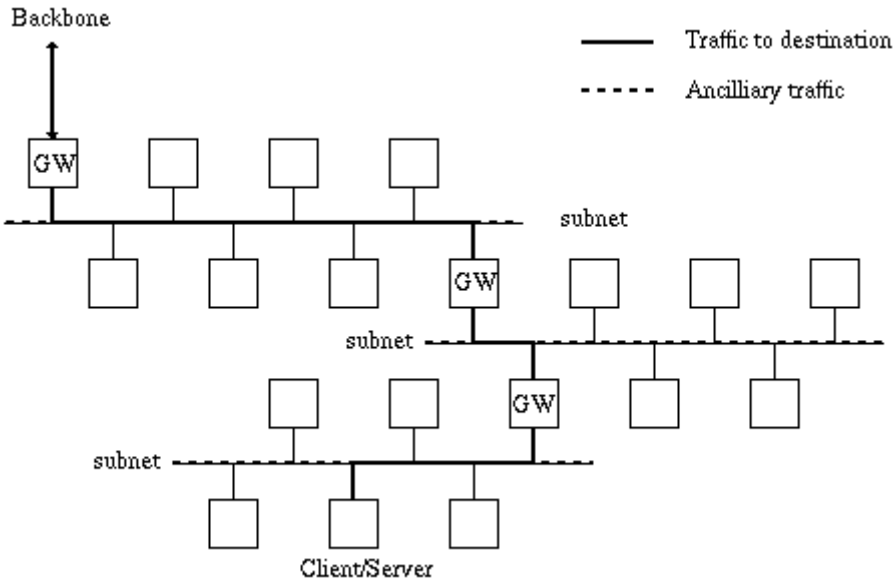

Figure 1. LAN Internal/External Traffic

In the case of figure 1, we see that a client requesting a document generates LAN Internal Traffic that could be avoided by proxying in an internal gateway such as with CERN proxy, or by browser caching, such as the one used by Netscape. These two schemes, depending on the specific page being accessed, may also reduce WAN Single and Multi Source Traffic.

The same computer, serving a document generates LAN External Traffic which can be avoided by means of server side caching at the gateway between the LAN and the backbone or better, even further down, if a cache is known to exist closer to the client.

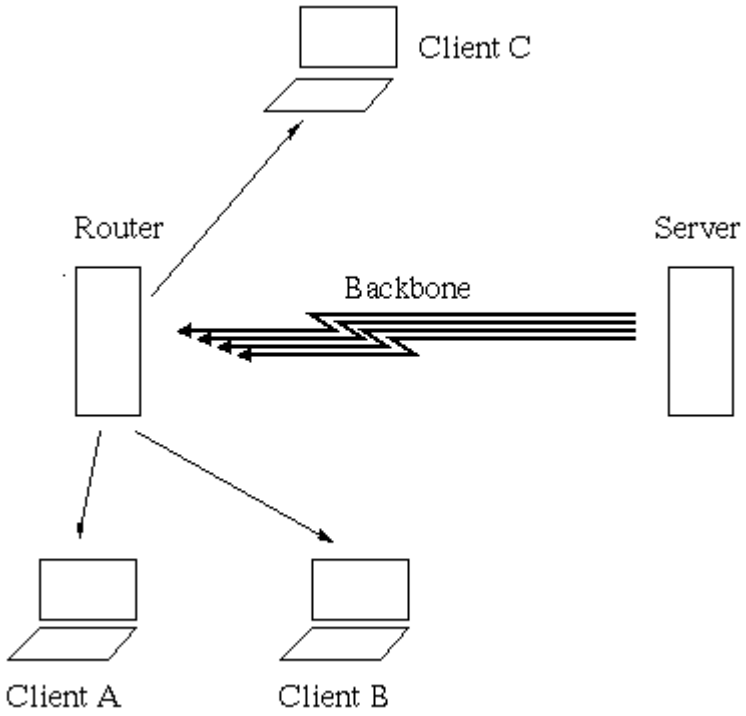

Figure 2. WAN Single Source Traffic

A multi-collaborative cache for the World Wide Web

Overview

Our proposal is designed to obviate LAN External Traffic and WAN Single and MultiSource Traffic, and it is compatible with Netscape and CERN type proxying. As well, it takes advantage of the CERN supported **If-Modified-Since** modifications.

The server only forwards clients to caching servers which are known to hold fresh copies. At the caching side, copies are held as long as indicated in a header of each file served or until the space is needed, whichever occurs first. As the server knows the vital statistics of each file (such as size, last modification date, and access frequency, and in some instances the expiry date of a document), it can automatically place a expiry date derived from these figures using a formula of the form

Expiry Date = (Today)+(Frequency of Accesses)+(Time since last change) +(Size);

where each of the terms might be weighted in an appropriate form.

The proposed protocol is somewhat akin to proxies, most of which have been investigated in terms of physical proximity be it in LANs or slightly wider geographical areasBes95, Gwe94, Gwe95]. To our knowledge, none of this proposals have gone beyond a detailed description of the problem. However, those studies provide valuable data for the design of a caching protocol.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.